AFRL-RI-RS-TR-2013-039

FAULT TOLERANCE FOR FIGHT THROUGH (FTFT)

FEBRUARY 2013

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2013-039 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /

JAMES PERRETTA
Chief, Cyber Assurance Branch

/ S /

WARREN H. DEBANY, JR.
Technical Advisor, Information Exploitation & Operations Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
FEBRUARY 2013	FINAL TECHNICAL REPORT	NOV 2009 – NOV 2012

4. TITLE AND SUBTITLE

Fault Tolerance for Fight Through (FTFT)

5a. CONTRACT NUMBER
IN-HOUSE

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)

Kevin Kwiat

5d. PROJECT NUMBER
23G4

5e. TASK NUMBER
IH

5f. WORK UNIT NUMBER
01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RIGA
525 Brooks Road
Rome NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RIGA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2013-039

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2013-0584
Date Cleared: 7 Feb 2013

13. SUPPLEMENTARY NOTES

14. ABSTRACT

When our cyber defenses' ability to prevent, avoid, and detect an attack are outmaneuvered and our information systems face impending loss of critical services, a fight-through capability must remain; otherwise restoration of those services may come too late for us to emerge undefeated. The task of "protecting the protector" drives us to create a fight-through capability that is hardened and heavily defended in cyberspace; however, these attributes alone are a "Maginot Line" that begs the question of why cyber attacks succeed in the first place. The more realistic goal is to design a fight-through capability that can absorb punishment and then rebound so that it can be the basis for restoration of critical services. Adaptations of fault-tolerant computing concepts have been applied to address needs in cyber defense. We likened the fight-through problem to an Observe, Orient, Decide, and Act (OODA) loop. Redundancy, as the underpinning of fault tolerance, was strategically placed to counter an attacker's optimal strategies. The fight-through OODA loop was aimed to outperform the adversary's OODA loop.

15. SUBJECT TERMS
Cyberspace, fault-tolerant networks, fight-through, network security, survivability, on-line social networks, sensor networks, cognitive radio

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**KEVIN A. KWIAT**
U	U	U	UU	50	19b. TELEPONE NUMBER (*Include area code)* **(315) 330-4574**

TABLE OF CONTENTS

List of Figures i
List of Table iii
Acknowledgments iii
1.0 Summary 1
2.0 Introduction 2
3.0 Methods, Assumptions, and Procedures 4
3.1 Replicated Computer in a Cluster Environment 5
3.2 Game Theoretic Formulation of Defensive Strategy Against Rational Attackers 7
3.3 Optimal Resource Allocation for Protecting System Availability 8
3.4 Continued Research in Voting Algorithms 9
3.5 A Brief Examination of Software Diversity: From Fault to Attack Tolerance 10
3.6 Replication and Diversity: A Game Theoretic Approach 18
3.7 Trusted OSN (Online Social Network) Services 24
3.8 Dynamic Reconfigurable Routing for Wireless Sensor Networks 28
3.9 Bargaining for Radio Spectrum Sharing 31
4.0 Results and Discussion 35
5.0 Conclusions 37
6.0 References 39

List of Figures

Figure 1: A Basic Observe, Orient, Decide and Act (OODA) Loop 1
Figure 2: Computer Clusters with Random Dictator and Random Troika 6
Figure 3: A Sample Scenario for Diverse Code Execution 15
Figure 4: Repeated Game Simulation with Five Nodes..... 22
Figure 5: Hierarchical Troika Nodal Arrangement..... 24
Figure 6: States of Data Sharing on OSNs..... 26
Figure 7: Dynamic Routing Framework 30
Figure 8: Pareto Optimal SPNE of the Respective Offerer in Different Periods..... 34
Figure 9: Concentric OODA Loops: Attacker's Outer Loop, Defender's Inner Loop 37

List of Tables

Table 1: Cyber Game Classification 19
Table 2: Combination of Preamble Bits and Their Inferences 29

Acknowledgements

This research was performed with support from the Air Force Office of Scientific Research under the auspices of the Laboratory Research Independent Research (LRIR) Program.

1.0 Summary

When our cyber defenses' ability to prevent, avoid, and detect an attack are outmaneuvered and our information systems face impending loss of critical services, a fight-through capability must remain; otherwise restoration of those services may come too late for us to emerge undefeated. The task of "protecting the protector" drives us to create a fight-through capability that is hardened and heavily defended in cyberspace; however, these attributes alone are a "Maginot Line" that begs the question of why cyber attacks succeed in the first place. The more realistic goal is to design a fight-through capability that can absorb punishment and then rebound so that it can be the basis for restoration of critical services. Adaptations of fault-tolerant computing concepts have been applied to address needs in cyber defense.

Military strategist John Boyd conceived and developed the Observe, Orient, Decide, and Act Loop (OODA Loop) [1]. He applied the OODA Loop to the combat operations process including the engagement of fighter aircraft in aerial combat. Figure 1 shows a basic OODA Loop.

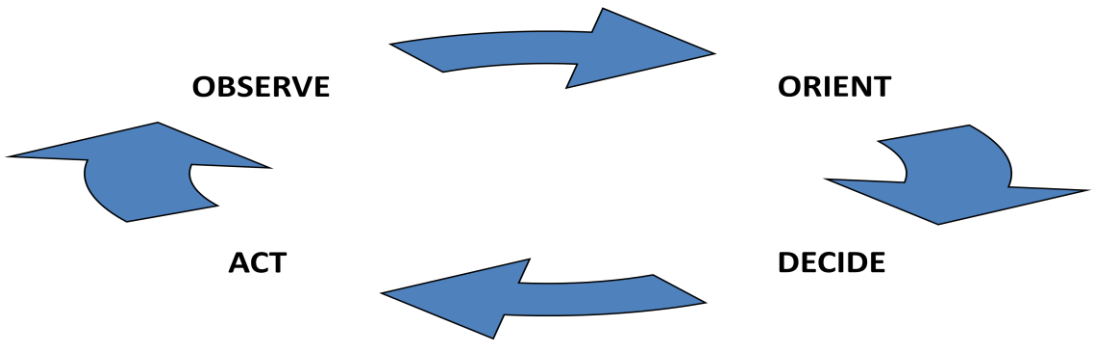

Figure 1: A Basic Observe, Orient, Decide and Act (OODA) Loop

We likened the fight-through problem to an Observe, Orient, Decide, and Act (OODA) loop. Redundancy, as the underpinning of fault tolerance, was strategically placed to counter an

attacker's optimal strategies. The fight-through OODA loop was aimed to outperform the adversary's OODA loop.

Our goal was to discover and deliver new ways for adapting concepts from the field of fault-tolerant design to create a cyber fight-through capability that absorbs punishment from attacker-induced faults yet prevails. Spatial, temporal, and information redundancy are the dimensions that, when stood-up concurrently, can withstand loss of resources in one dimension while the other dimensions supply the reinforcing resources for a fight-through capability. Fault tolerance for fight through (FTFT) spans multiple dimensions of redundancy to form an OODA loop for fighting-through. By being able to *observe* an attacker's attempts to create faults in one dimension, FTFT can *orient* the other dimensions and then *decide* on their deployment in order to *act* against the attack – by fighting through it.

2.0 Introduction

Fault tolerance in information technology and computing share several conceptual similarities with cyber defense. For example, fault tolerance deals with detection and treatment of failures and cyber defense deals with detection and treatment of violations. A well-formed and well-founded framework for treating violations comes from fault-tolerant computing. Traditionally, fault-tolerant computing dealt with randomly occurring faults and not faults results from intelligent attack. However, a reliable system should be survivable against the faults caused by cyber attacks as well as the internal failures. Whereas faults caused by natural-occurring phenomena are tolerable using established, standard approaches, attacker-induced faults require a more aggressive approach. New challenges arise in the area of transforming fault tolerance to attack tolerance.

We have investigated others' research into similar uses of fault tolerance. DARPA's Organically Assured and Survivable Information Systems (OASIS) program performed and documented [2] research that aimed to develop "…ways to enable critical DoD computers to operate through a cyber attack, degrade gracefully if necessary, and allow real-time

controlled trade-offs between system performance and system security through such techniques as redundancy and diversity of operating systems." This research has applications in robust, self-forming networks required to successfully conduct network centric warfare.

Based upon the results and insights from the OASIS program, DARPA launched the Self-Regenerative Systems (SRS) Program [3]. The SRS program, like the OASIS program, sought to adapt concepts from fault-tolerant computing to cyber security. One notable objective of the SRS program was that it aimed to surpass the fault tolerance attribute of graceful degradation set forth by the OASIS program. The SRS viewed graceful degradation as a continual depletion of resources that would eventually lead to the defeat of information systems; instead, the SRS program sought to have information fully-rebound from attacker-induced faults. Furthermore, given experiential knowledge of the attack, the information system could be immune to that attack in the future. This learning technology was coined *cognitive immunity*. Encountering the same attack again would not adversely affect the information system; instead, once they became capable of defeating the attack, self-generative systems were aimed at exceeding previous levels of performance.

If fault tolerance is to be a trusted basis for a fight-though capability, then we must anticipate that the fault tolerance mechanisms themselves will be subjected to unanticipated conditions created by attackers. Anticipating the onset of the adverse effects of an attack is the subject of research in estimating an information system's mean time-to-compromise (MTTC) [4]. Related research into estimating the mean-time-to-breach (MTTB) [5] analyzes an attacker's behavior. Although the terms "breach" and "compromise" are similar, measuring MTTB requires significant detail about the target system whereas MTTC is measured in a comparative way using observable variables rather than a calculated indicator. The work in estimating a MTTC is used by security architects and managers to intelligently compare systems and determine where resources should be focused. Like MTTB, MTTC estimates require variables that are difficult to exactly quantify such as attacker skill levels. MTTC and MTTB are used to deploy resources. Some of these resources may be redundant, such as redundant data for information recovery or redundant services for continuity of operations.

Redundancy plays a fundamental role in fault tolerance, since specific resources must be in place to deal with a fault when it is encountered. Spatial, temporal, and information redundancy are used to create fault-tolerant systems [6]. Spatial redundancy is the physical replication of hardware and software components. Information redundancy is the addition of redundant information or data. Temporal redundancy attempts to reduce spatial and information redundancy at the expense of using additional time. Designers of fault-tolerant systems must understand the types of redundancy techniques available and the methods that can be used to evaluate the impact of the redundancy. Fault-tolerant system designs will often combine redundancy types. For example, spatial and information redundancy are used together in Redundant Area of Independent Disks (RAID). RAID provides survivable storage where the data is striped across multiple disks (spatial redundancy) and parity bits (information redundancy) are added.

3.0 Methods, Assumptions, and Procedures

The technology that we developed includes models, algorithms, and protocols to ensure that redundancy for fault tolerance creates an OODA loop for fighting-through. An important step in this direction is maintaining consistency among replicas. We therefore begin this discussion of our methods, assumptions, and procedures by choosing a telling example to illustrate the basis of our intended work. The example involves voting among replicated computers. The process of replication in information systems forms a basis of system survivability. Replication can overwhelm an attacker with too many targets, while also ensuring that a sufficient number of components will survive the attack. When an attacker breaches the defenses that surround certain system components and subsequently overtakes those components, their replicas carry out the mission. The premise behind replication as a defense mechanism then is that although the system's internal resources have been diminished, it survives. A mechanism, however, is merely a trigger; procedures are used in conjunction with the mechanism to take the actions to fight through the attack. For these procedures we use game theory.

Section 3.1 introduces the use of replicated computers for a fight through capability and sets the stage for using game theory. Section 3.2 elaborates on the application of game theory in formulating a defensive strategy for replicated computers. Section 3.3 extends the defensive strategy to specifically preserve the availability of service provided by the replicas. Section 3.4 describes our recent advance in resolving the replicas' redundant outputs, and this leads to Section 3.5 where we consider the how the replicas' concurrent tasks impact the use software diversity. Building upon this foundation of combining replication with diversity, Section 3.6 describes a game-theoretic approach for fighting through attacks. Section 3.7, marks a departure from the predominate theme of fault tolerance. In Section 3.7 we examine trust in on-line social networks. Section 3.8 describes a new approach for routing in wireless sensor networks. An improved method for sharing of spectrum between cognitive radios is given in Section 3.9.

3.1 Replicated Computers in a Cluster Environment

Consider a cluster of computers where voting is performed. Each of the computers is redundant in the sense that they perform an identical task that is replicated among them all. Some computers in the cluster may be fault-free while others may be faulty. The most-often applied assumption is that a majority of the computers are non-faulty, so the voting decision is formed by majority rule. What drastically modifies this assumption is that faults may be attacker-induced.

At the onset of our research we dealt with two noteworthy attack-tolerant systems: *random dictator* and *random troika*. Figure 2 shows 2 clusters of 9 computers each: one cluster employs random dictator and the other random troika. Note that the majority of 9 is 5, so majority voting among all the computers within a cluster would tolerate, at most, 4 failed computers. Random dictator and random troika represent different voting schemes. In Figure 2, the random troika only considers majority voting among 3 of the 9 computers within a cluster. Random dictator really performs no voting at all; instead, a computer selected at random "dictates" that its result should represent all of the computers in its cluster.

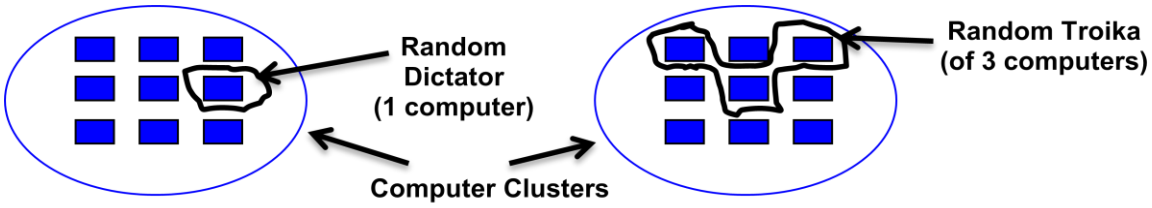

Figure 2: Computer Clusters with Random Dictator and Random Troika

A published [7] FTFT result shows that the added fault tolerance of majority rule is apparently trumped by the added attack tolerance of the random dictator. Although it is counter intuitive to create a single point-of-failure, with random dictator, the whole purpose of voting in the first place (to the extent that the random dictator system would even be considered "voting," in the everyday use of this word) is shifted away from its original one of replacing individual judgments with presumably more reliable collective judgments. In random dictator, the purpose of voting becomes that of creating a shell game hiding the identity of the dictator from the enemy. In the more formal game-theoretic analysis of this contest between defender and attacker, the Nash equilibrium solution concept suggests that the attacker will continue to concentrate his forces so as to corrupt a bare majority of the computers in as many clusters as possible, even if he suspects that the defender will use random dictator, against which no attack strategy is more or less effective than any other. The reason for this behavior is to deny the defender the use of the otherwise more attractive majority rule system. One effect, then, is that *by concentrating his forces, the attacker denies the defender the benefit of fault tolerance lent by majority rule*. Therefore, if our policy for cyber defense were to adopt a purely random dictator approach, then we inadvertently give a strategically-thinking attacker an edge by becoming highly vulnerable to naturally-occurring faults.

To fight through, we could resort to something other than random dictator that combines a

degree of fault tolerance with some of the random dictator's shell-game effect. It seems possible that randomly selecting a small subset of voting computers (called a random troika - see Figure 2 above) in the cluster would provide a better defense than both random dictator and majority rule for some examples similar to the one we have been discussing. If the attacker continued to corrupt a bare majority of the computers in as many clusters as possible, then random troika would out-perform majority rule in the corrupted clusters, but in these clusters it would not do as well as random dictator. In the unattacked clusters, random troika would out-perform random dictator, and be out-performed by majority rule. But such an analysis fails again to take account of strategic factors; if random troika were available as a defense option, then the attacker's optimal strategy might no longer be to corrupt a bare majority of the computers in as many clusters as possible. Certainly, it then became necessary to do a careful strategic analysis with random troika added as an available fault tolerance-based defense strategy.

3.2 Game Theoretic Formulation of Defensive Strategy Against Rational Attackers

Motivated by the consideration of strategic factors posed by deploying the random troika, we developed an optimal voting strategy (i.e., the optimal number of participating voters) against rational attacks whose goal is cause total failure by strategically compromising individual voters across the system. According to Myerson [8], "game theory can be defined as the study of mathematical models of conflict and cooperation between intelligent rational decision-makers". Myerson's notion of "rational decision-makers" takes on a malicious connotation when applying game theory to cyber attacks and the highly unconventional tactics such attackers can employ. Simply put, rational attackers are ones whose actions do not deliberately diminish the chances to achieve the attackers' goals. Armed with this notion, we modeled [9] the problem of deciding the number of participating voters against rational attackers as a two-person zero-sum game problem and provided solutions based on the results of this well-known game problem. A set of experiments was performed to illustrate the devised majority voting strategy when varying numbers of replicas are deployed and undergo aggression from rational attacks. The aggregate replicas' ability to maintain the integrity of their majority output changes when the individual replicas' reliability changes

and when the number of compromised replicas grows. We found the optimal number of participating replicas participating in a majority vote when the system undergoes rational attacks, and we provide the solution for deciding the optimal number of voters that maximizes the expected number of clusters that will produce a correct result while the system is under attack. Three sets of experiments were performed investigating the relationship between the voting strategy, the attacker's strategy, and the ability of the system's fault tolerance to fight through the attack and preserve overall system integrity.

3.3 Optimal Resource Allocation for Protecting System Availability

Whereas our work in [9] dealt primarily with spatial redundancy (i.e., replicated computers in a cluster environment), the use of multiple dimensions of redundancy – spatial, temporal, and information – became more apparent in our work [10] that focused on maintaining availability of the system's core services' in an adversarial environment. In our modeling of this problem, the defender seeks to maintain maximum system availability for a given period of time and does so by distributing defensive resources to attain the following: enhancing an individual replica's protection mechanisms; creation of additional replicas to overcome the attacker with too many targets; and producing camouflaged components that appear like functioning replicas but are merely decoys. With this environment, we see that redundancy for fault tolerance is used for fighting through. The spatial redundancy of replicas is accompanied by the information redundancy of the extra message traffic generated due to the camouflaged replicas. In a separate paper [11], we demonstrated how this form of redundancy burdens attackers who are attempting to probe the system by enmeshing them in an indiscernible pattern of message traffic that often misleads them from their intended target - thus lengthening the mean-time-to-breach a replica. Temporal redundancy is embodied in [10] by the repetitiveness of the camouflaged replicas where their seemingly meaningful efforts merely mimic the action's of the true replicas but at another time. These instances of information and time redundancy support our assumption that the attacker has no a priori knowledge about the system configuration; instead, the attacker has to gain this knowledge. The assumed inability to distinguish between fake and true replicas or to distinguish between replicas having enhanced protection and those without means that, at the onset, the attacker

resorts to random attacks on these replicas that are perceived to provide the system's core services. We formulated this attacker-defender problem as a defender's optimization problem and presented an algorithm that optimally allocates the above system resources to maximize the system's availability. Our analysis of the attacker-defender problem in which the defender's resources are limited and must be distributed between the three different approaches to protect system, (i.e., adding redundant, or camouflage replicas, or installing enhanced security scheme on existing replicas) led to our devising an algorithm that optimally allocates this triad of resource types so as to maximize the system's availability. Three sets of experiments were performed to investigate the relationship between: the triad of resource types and system availability; attack time and different resource allocation strategies; and resource allocation strategies and the number of replicas providing core services. In this work, however, we did not consider the cost the attacker anticipates accruing when promulgating the attack to different replicas in the next step of the game. If such costs are taken into consideration, the optimal solution may be viewed from two different perspectives: first, the attacker's will consider how frequently to switch to another targeted replica while striving to maximizing the system damage inflicted; second, the defender will analyze the attacker's strategy first and then take those countermeasures deemed necessary to minimize the system damage. In the future, we plan to apply game theory to this strategically-rich scenario; however, in Section 3.6 we apply game theory to form a replica's reputation and use this reputation to assist in fighting through an attack.

3.4 Continued Research in Voting Algorithms

Prior to the FTFT effort, we had conducted research into voting algorithms; yet, as voting is oftentimes inseparable from the deployment of replicated computing resources to fight-through cyber attacks, we carried out additional research in the area. Our work in FTFT employed hierarchical adaption methods to manage the Quality-of-Service (QoS) at various levels of a replica voting system: namely, the timely delivery of correct data to the end-user. The fine-grained adaptations occurring in the voting system are controlled macroscopically via the QoS reconfiguration actions occurring at the application layer, in a context of the external events sensed by a situational assessment module. Whereas our past work had

focused on engineering the voting protocol mechanisms to lower the time-to-complete (TTC) a voting round and the bandwidth consumed therein, our current work [12] considered the dynamic nature of the network bandwidth availability to support the voting operations and the unpredictable changes in device-level fault occurrence and network message loss and delay. We treat the voting system as a "black box" with known I/O behaviors, which is then exercised by the situational assessment module for macroscopic control. Our hierarchical adaptation approach offers the potential to keep the data miss rate of the voting system within the allowed limits as the system and environment parameters change.

3.5 A Brief Examination of Software Diversity: From Fault to Attack Tolerance

High assurance of computing and information systems is challenged by the very entity that makes such systems possible – software. This is because software is especially susceptible to unanticipated faults. The Apollo guidance computer offers a historical example. Technologists claim that the goal to put a man on the moon propelled the advancement of integrated circuits (ICs) from what were nascent prototypes to proven chips having full-scale manufacturability [13]. The capability for sustained production of reliable ICs launched the 3rd generation of digital computers. However, in contrast to being credited with helping reach this milestone of hardware development, the U.S. Air Force noted that the Apollo lunar missions, in spite of being one of the most carefully planned and executed software projects ever taken, attributed nearly all their major problems to software design faults [14]. The design successes of the Apollo space borne computer reside predominantly in hardware. Since then, the overwhelming concern for design faults had thus not been directed towards hardware; instead, design faults consistently became a major concern in software [15]. Indeed, the major portion of a system's complexity is to be found in the software, and design faults naturally stem from complexity.

Software faults are design faults – introduced when the software is created. They can occur throughout the design process including programming. If they escape detection during the testing and debug stages, then these faults remain within the fielded software. Recognizing the potential significance of such faults, techniques to tolerate them have been proposed for

high assurance systems. Fault tolerance is almost always enabled through the use of redundancy; yet, unlike physical faults, software faults occur only at design time, so the ability to tolerate them was sought through redundant – albeit different - designs. In the 1970's, two main techniques for tolerating software faults emerged. They are the Recovery Block (RB) scheme [16] and *N* Version Programming (NVP) [17]. In general, these schemes employ multiple variants of the software in an attempt to guarantee that at least one variant will pass the correctness checks that are performed while the software executes. The checks include voting on results that may not exactly agree, but are nonetheless correct, and determining the reasonableness of a variant's results. One study [18] advocated RB and NVP for creating a combined hardware-software fault-tolerant architecture because the checks these schemes employed were deemed sufficient to detect hardware faults as well. However, the establishment of quantified estimates of the improvement in system reliability that could be expected from using RB or NVP never fully materialized [15]. Without measurable benefits to be derived from the creation of multiple software variants, software fault tolerance did not become widely accepted; instead, it has been limited to extreme cases such as the NASA Space Shuttle's redundant computers. During the Shuttle's flight-critical phases, one computer ran independently-designed software apart from the other computers' lock-step execution of identically-replicated software. This alleviated NASA's fears of a generic bug causing simultaneous failure of the redundant computers all running identical software [19]. The perceived, but nevertheless unquantified, gain in assurance justified the creation of diverse software.

Let us turn, for the moment, to more contemporary concerns and how software diversity has come to the forefront. Concerns over the absence of software diversity in modern computing and information systems spurred a study by Birman and Scheinder [20]. Being in a software monoculture, these systems can be perceived as being susceptible to a common flaw. The flaws, however, are not design faults that, when encountered by a running program, would cause the program to fail; instead the targeted flaws are those that create vulnerabilities in the software that are exploitable to *attack*. In such an environment, an attack that successfully exploits a single global vulnerability could compromise all machines of the monoculture. Addressing this vexing problem, Birman and Scheinder defined 3 types of attacks that could

be directed at a monoculture, and for each type, they outlined the corresponding defense [20]:

- **Configuration:** Exploit aspects of the configuration. Vulnerability introduced by system administrator or user who installs software on the target. **Useful Defense:** Monoculture such as Standard Desktop Computer

- **Trust:** Exploit assumptions made about the trustworthiness of a client or server. Vulnerability introduced by system or network architect. **Useful Defense:** Principles of least privilege and/or formal methods

- **Technology:** Exploit programming or design errors in software running on the target. Vulnerability introduced by software builder. **Useful Defense:** Diversity

Flaws are at the underpinnings of all these attacks. For example, when systems are misconfigured, confidentially of the system could be compromised by a configuration attack that successfully skirts the system's encryption. A trust attack could result from a flawed assumption – one not rigorously substantiated. For example, a server considered as furnishing trusted information may actually have good information supplanted with bad - thus leading to a violation of information integrity. Apart from unintentional software flaws manifesting as software faults that could cause a program to fail, fault-free code becomes detrimental to a program – or flawed - when it enables attack. Specifically, the *technology*, in what could very well be a fully-functional program, is maliciously manipulated by the attacker so that the program goes into behavior that is unintended from its design.

Misconfigurations are adverse to a monoculture because a monoculture's components operate in a uniform manner, and when actions are done frequently and repeatedly, they become less prone to error. Enforcing the principles of less privilege and applying formal methods avoids bona fide behaviors that are oblivious to the pitfalls of misplaced trust. Stemming from diversity of software is a multiplicity of underlying avenues for the attacker to attempt malicious manipulation; yet the multiplicity forms a defense. Instead of offering

the attacker a richness of targets, the attacker now has to contend with temporal and spatial barriers: the attacker needs enough *time* to successfully attack a sufficient *number* of distinct software targets.

High assurance of computing and information systems – especially those on enterprise scale that would entail the embracing of a monoculture – calls upon software diversity. We have seen that software diversity for high assurance has spanned over half a century of computer history. During that time, software diversity did not always garner high acclaim. In spite of its longevity, a period even existed when software diversity was viewed as a detriment to high assurance. In 1990, Abbott argued that software fault tolerance techniques, as represented by the RB and NVP techniques, are departures from sound software engineering practices [21]. In addition to the high expense for RB's and NVP's multiple software variants, Abbott took exception to the notion that software fault tolerance's capturing of software design faults in the field is somehow better than devoting sufficient engineering resources to detecting the same faults during normal program testing. He further noted that in order for either RB or NVP to tolerate software design faults, a variant of the software that handles each of the incorrect but tolerated cases must be designed and implemented ahead of time. The acceptability of releasing potentially faulty variants alongside those that are supposedly fault-free served as empirical evidence that a thorough design-and-test methodology had been prematurely and deliberately abandoned. Whatever design faults remained were to be handled, by RB or NVP, in the field. Abbott wryly referred to this situation as "delayed debugging ".

Adding to the perceived futility of RB or NVP, Abbot noted their complete dependence on the traditional design cycle to provide the code for a variant to do the correct processing. He proclaimed [21], "It certainly is not the case that when a fault appears the system dynamically generates new corrected code!" This highly dismissive stance, although soundly argued, is now viewed more promisingly when the system being considered is confronted with attacks. Software diversity that is not merely statically created at design time but is dynamically generated by the system strengthens the system's defense.

In 2000, DARPA assessed the existing approaches to information system security and survivability – tenants of high assurance - as consisting of preventing, detecting and containing unintentional errors and attacks. These systems, by employing static means for high assurance, largely mimicked principles from fault-tolerant computing. The realization that such systems were too vulnerable to attacker-induced faults gave DARPA the impetus to initiate the Organically Assured and Survivable Information System (OASIS) program [2] in partnership with the Air Force Research Laboratory's (AFRL's) Information Directorate. The OASIS program was a cross-disciplinary program that combined fault tolerance and Information Assurance (IA) technologies to build information systems that detect, contain and operate through attacks in the then emerging cyberspace. Following the OASIS program's successes, a challenge of operating through attacks emerged: resource depletion. Regardless of how well systems are protected or how well they tolerate errors and attacks; they will eventually fail over time unless they have the ability to replenish lost resources. On the positive side, OASIS-type systems that tolerate attacks by gracefully degrading services afford time for these systems to regenerate. In 2004, DARPA began funding work in Self-Regenerative Systems aimed at replenishing themselves of lost due to unforeseen errors or attacks and automatically improving their ability to deliver critical services. The sought-after product would make information systems and data persistent. Such information systems must have redundancy and the ability to regenerate required functionality with increased error and attack immunity, whereby corrupted components can be regenerated without negatively affecting the whole system. FTFT intended provide a complementary effort that was to expand replication from being merely large-scale deployments aimed at foiling an attacker with too many targets. By introducing dynamic diversity in runtime environments, FTFT sought to invoke a machine's ability to automatically recover from attacks – demonstrating the passage from fault to attack tolerance.

The complexity of regeneration warrants an illustrative, albeit simplified, example for survivable voting execution with software diversity. Figure 3 shows 3 host computers, A, B, and C, each executing diverse, but functionally-equivalent versions of code. The code is logically divided into stripes such that when a stripe boundary is encountered, A, B and C produce output, in the form of votes, to a poller P. Assume that the host's vote represents the

state of that host. Since the code of each of the stripes is diverse, the votes must be allowably different yet still be correct; therefore, the poller must be able to perform inexact comparisons between the votes to determine if there is a disagreement among them. Figure 3 shows that host C, being a victim of an outside attack, casts a disagreeing vote. The poller in this case is able to create a majority vote that represents the uncompromised computers A and B. If the code in all of the 3 computers were completely identical, then regeneration would be straightforward, but by introducing diversity, any necessary of translation of data and state that underlies the regeneration is assumed to be performed by the poller.

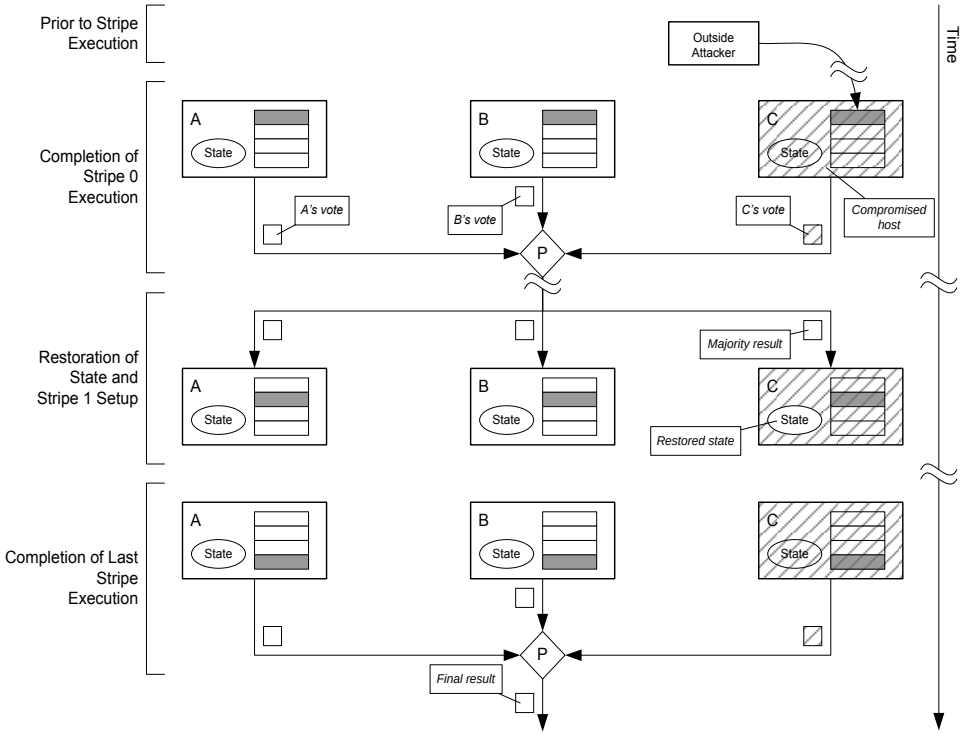

Figure 3: A Sample Scenario for Diverse Code Execution

This is a simple example of corrupted components – the state of the code in this case - being regenerated. However, the stripe executions in Figure 3 shows that the regeneration does not endure; instead, host C promulgates the attack by corrupting its state as illustrated in the last stage of Figure 3. Although no harm is ultimately done because of the poller's ability to mask host C's votes, the incompleteness of the regeneration puts this system on the brink of losing

its assurance. If either machine A or B were to also become compromised, then the final result produced in Figure 3 is corruptible. From this perilous position, A and B must unify before another compromise precludes any possibility of further regeneration. A complete regeneration would eliminate host C and replace it. In realizing a rebounding from the attack with a replenishment of the previously-lost computer C, A and B must act in unity – agreeing, say, on the admission of a new member to the triumvirate. Such an agreement, even in the simple collection of computers of Figure 3, involves performing the act of consensus. Seeking high assurance of regeneration incited an investigation into combining software diversity with consensus protocols to form an attack-tolerant system.

Instantiations of software diversity have been demonstrated through the automatic generation of code from a formal verification system [22]. In [22], formal verification in the combination of software diversity with consensus protocols underscores high assurance in making the passage from fault to attack tolerance. Abbott's skepticism of software diversity dismissed the idea that when a fault appears the system does not dynamically generate new corrected code. For attack tolerance, such faults would be considered as attacker induced; yet software faults in the classical sense are not to be ignored. Currently in [22], the regenerated code that is diversified to thwart sustained technology attacks is formally verified. Unlike being characterized as "delayed debugging" the diversity in [22] is grounded in rigor. This avoids the common pitfall encountered when introducing new technology into a high assurance setting: susceptibility to faults is not inadvertently increased.

High assurance is also a compelling reason to avoid "delayed detection" of attacks. If evidence of attack is to be found in a computer's state, then it compels us, in the context of Figure 3, to frequently compare the computers' states to detect such evidence. Consensus and voting are inextricably bound: for participants to reach agreement they need to vote among themselves. The onerousness of "delayed debugging" stems from, according to Abbott, the undue burden it places upon fielded systems. He asserts that transferring the need to find and remedy faults from the design phase to when the system is in operation is not an efficient use of resources; instead, resources, such as time, could be allocated much more propitiously.

Performing comparisons, as in Figure 3, requires time. The act of comparison is a common ingredient to both consensus and voting, so it is readily available as a means to detect attacks during these operations. A tendency to "delay detection" could spell defeat because it may allow the attacker sufficient time, before a state comparison is made, to successfully attack a majority of the computers. Recalling the diverse code execution depicted in Figure 3, the actions of the poller are similar to those of a voter: the redundant outputs are compared in majority-voting fashion in order to mask the erroneous outputs of a minority. However, we used the term "poller" in Figure 3 instead of the term "voter" because a poller actively solicits outputs from the computers whereas a voter's normal operation is to passively wait for the computers to provide their outputs as the voter's input. A voter would only receive an input when the computers' programs are generating output – and this may be infrequent. The action of a poller, unlike that of a voter, is to frequently seek erroneous computer outputs as an indication of a successful attack corrupting a computer's state. The polling of the computers in Figure 3 is through the actions of the code stripes. In this situation, an attacker alters a computer's signature of a state (herein referred to as merely "state") when compromising the computer. The state could, for example, be chosen from a combination of some selected values of the diverse code and from indicators from each of the individual redundant computers' defenses. Instead of voting when the computers have outputs ready, the polling on a stripe boundary forces the computers to provide their intermediary results for comparison. Striping in this manner protects a computer by imposing a bound, *S*, on the stripe size. Since each computer's state is refreshed between stripe executions, no minority's state is preserved from one stripe to the next. Thus, a successful attack must be placed within a single stripe. This places an upper bound on the time of a successful attack of *S* machine instructions. Therefore, a stripe of size *S* will successfully thwart all attacks having the same temporal duration. Consequently, as *S* is decreased, the level of protection is increased. An immediate drawback to decreasing S is the increase in polling and the subsequent overhead of majority state regeneration. A subtle drawback is to diversity itself: striping, we contend, opposes diversity because of decreasing S. If voting is performed only when the computers have outputs ready, then the output specification for the diverse, but functionally-equivalent versions of the code would allow for comparison of the computers' outputs. Even if the

outputs are different but still conform to the output specification (due to allowable fuzziness in the output specification), then a form of inexact voting would permit these different-but-still-correct outputs to not be inadvertently flagged as erroneous. When the size of *S* is not aligned with the times that the code's output is created, then comparison of the output from the diverse, but functionally-equivalent versions of code must be purely on the internal state of the computers running the code. This need to specify the states so as to make them comparable is a hindrance to diversity because it forces a convergence towards a code monoculture among the computers. As *S* decreases so must the diversity of the versions of code. In the lower limit, when $S = 1$ the comparison is reduced to single - and thereby *identical* - instructions. In this case, striving for high assurance has ushered in a monoculture.

The notion that code diversity is limited in scenarios where replicated computers operate concurrently to fight through cyber attacks compels us to be mindful that code diversity will not be abundantly available. Therefore, in the next section dealing with game theory in the context of replication and diversity, we have taken a conservative approach to the degree of diversity that we assume is available for replicated computers.

3.6 Replication and Diversity: A Game Theoretic Approach

Game theory is the branch of applied mathematics that analyses *conflict* and *cooperation* among rational agents in strategic interactions. A strategic interaction is any interaction in which the behavior of one agent affects the outcome of others. Cyber defense comprises numerous strategic interactions. First, the attacker's behavior must affect the defensive strategy. Second, several protocols and security policies cannot be unilaterally implemented because they require the collaboration of several users or several organizations to be successful. Finally, cyberspace is interconnected and the data collected from one vulnerable computer (organization) can be used to compromise the others. In fact, using the framework of game theory, the network defender optimizes his resources and defensive strategy while taking into account the users' behaviors, other organizations, and the different attackers' actions. We created Table I to summarize the different classifications of cyber games.

Table I: Cyber Game Classification

Questions	Ans-wers	Types of Game	Remarks
Are the rules of the game already in places?	Yes	Game theory model	
	No	Mechanism design principle	
Are the players rational?	Yes	Game theory model	
	No	Evolutionary game model	Population of players, replicator dynamic evolutionary stable strategy
Can the contract or agreement between the players be enforced?	Yes	Cooperative game	Solution concepts: Core, Kernel, Nucleolus, Shapley value
	No	Non-cooperative game	Solution concepts: Nash equilibrium
Does the payoff depend only on the strategy and not the identity of players?	Yes	Symmetric game	
	No	Asymmetric game	
Does a player can benefit only at the equal expense of others?	Yes	Zero-sum game	Frequent in military application, pure conflict
	No	Non zero-sum game	Frequent in civilian application, opportunity of cooperation for mutual benefit
Is all players moving simultaneously or are later players not aware of earlier player move?	Yes	Simultaneous game	
	No	Sequential game	
Is all players knows the moves previously made by all other players?	Yes	Perfect information	Only sequential game can be of perfect information
	No	Imperfect information	
Is every player knows the strategies and payoffs available to the other players?	Yes	Complete information	
	No	Incomplete information	
Does the game have finite number of players, moves, events, outcomes?	Yes	Discrete game	
	No	Continuous game	Differential game
Is the game static or one-shot?	Yes	Static game	
	No	Dynamic game (see A)	
(A) Is the same stage game repeated?	Yes	Repeated game (see B)	
	No	Stochastic game	
(B) Does the players have perfect observability of others' past action	Yes	Perfect monitoring game	
	No	Imperfect monitoring game (see C)	
(C) Is the signal of past plays, however imprecise and noisy, invariably observed by all players?	Yes	Imperfect public monitoring	Players' signal perfectly correlated
	No	Imperfect private monitoring (see D)	Players' observe different signal of past plays. In the extreme case, players' signals are conditionally independent
(D) Do players, in their selfish optimization, need to infer the private history of other players based on their own imperfect observation?	Yes	Belief based equilibrium	
	No	Belief-free equilibrium	Easily tractable

We contend that game theory is a mature theoretical framework that enables the modeling of several realistic scenarios.

The research in [23] examines cyber defense – particularly those technologies that target cyberspace survivability. An effective defense-in-depth avoids a large percentage of threats and defeats those threats that turn into attacks. When an attack evades detection, is not defeated, and disrupts systems and networks, the defensive priority turns to survival and mission assurance. In this context, mission assurance seeks to ensure that critical mission essential functions (MEFs) survive and fight through the attacks against the underlying cyber infrastructure. Survivability represents the quantified ability of a system, subsystem, equipment, process, or procedure to function continually during and after a disturbance. US Air Force systems carry varying survivability requirements depending on the MEFs' criticality and protection conditions. Almost invariably, however, replication of a subsystem, equipment, process, or procedure is necessary to meet a system's survivability requirements. Therefore, the degree of replication within a system can be paramount for MEF's survival. In fact, particular subsystem may fail, but the overall system survives because the functions performed by the failed component are replicated. We cautiously prescribe diversity to the replicas. They are assumed to be functionally-equivalent; yet their ability to generate comparable outputs does not inadvertently drive towards a monoculture. We assume that such diversity would stem from a randomization of some implementation aspects instead of having been based on systems that are differently designed. For example, the randomizing of system call names would be more conducive to attaining functionally-equivalent replicas rather than the diversity achieved through having each of the replicas' code designed by independent teams.

Among FTFT's research contributions, we developed a scheme that tracks a replica's history leading to the building of that replica's reputation – a measure of how much a defender can believe in that replica's genuineness. This approach uses a mechanism based on a repeated game. Specifically, a three-part mechanism - totally controlled by the defender - ensures high decision reliability through voting.

The first part of the game-theoretic mechanism (in the context of a replica voting

mechanism) is an exponentially weighted moving average to accurately update the node reputation according to the most recent behavior. Specifically, a node i reputation $R_i(t)$ at times t is updated according to the following recursive formula.

$$\begin{cases} R_i(0) = 0.5 \\ R_i(t) = (1-\gamma)R_i(t-1) + \gamma \text{ if node } i \text{ vote correctly} \\ R_i(t) = (1-\gamma)R_i(t-1) \text{ if node } i \text{ vote incorrectly} \end{cases} \quad (1)$$

γ is the smoothing factor, $0 < \gamma < 1$.

The second part is a mathematically proven optimum weight w_i derived from the node's reputation $R_i(t)$. Clearly,

$$w_i = log \frac{R_i(t)}{1 - R_i(t)}. \quad (2)$$

The vote weight in (2) takes advantage of misleading information from malicious nodes. For instance, if the defender knows that a specific malicious node lies all the time (e.g. the node has zero reputation, negative infinite weight). The information from that node should be inverted and used to get the true state all the time (100% sure). Equation (2) generalizes this concept when aggregating the vote from several nodes. Clearly, a node having a negative weight (or a reputation less than 0.5) has its binary vote flipped before computing the final result. Nodes with positive weight have their vote unchanged. A node with zero weight has its vote practically eliminated if at least one other node has a weight different to zero.

The third part is a game separation method that discourages malicious nodes to accumulate any reputation or have any weight in the decision process. When the discount factor δ is large $(\underline{\delta} \leq \delta < 1)$, the malicious nodes are tempted to accumulate a reputation for a potential future damage. To prevent this, the defender must divide the game using the framework originally proposed by Ellison [24]. With this framework, the defender divides the game in *M* separate games and record separate reputations for each game. The first game taking place in period *1, M+1, 2M+1,3M+1….* The second in period *2, M+2, 2M+2,…*, and so on. Since the games are separate, the outcome of one game does not influence the outcome of the other game; a malicious node's best response must be independent across the different games. As a result, the new discount factor in each of the separate game becomes δ^M, which

monotonically decreases as M increases. Therefore, there exists M_δ such that for all $M > M_\delta, \delta^M < \underline{\delta}$. The defender may choose M to be the least greatest integer such that M is greater than M_δ up to the time the last regular node is compromised. The more intuitive way to understand the mechanism we just described is that it will take a longer time for a malicious node to accumulate reputation in each separate game. Say, for instance, that the nodes vote once a day and possibly accumulate a reputation. If the defender divides the game into 365 parts of day-long duration, then a malicious node that accumulates any reputation today must wait a year before creating any damage. Therefore, after game separation, it becomes optimal for all compromised nodes to cast bad votes (play *N*) and not accumulate any reputation at all. As a consequence, because regular nodes have a higher weight in the aggregate decision, the aggregate vote reliability stays above 50% even though nearly all the nodes are compromised (see Figure 4).

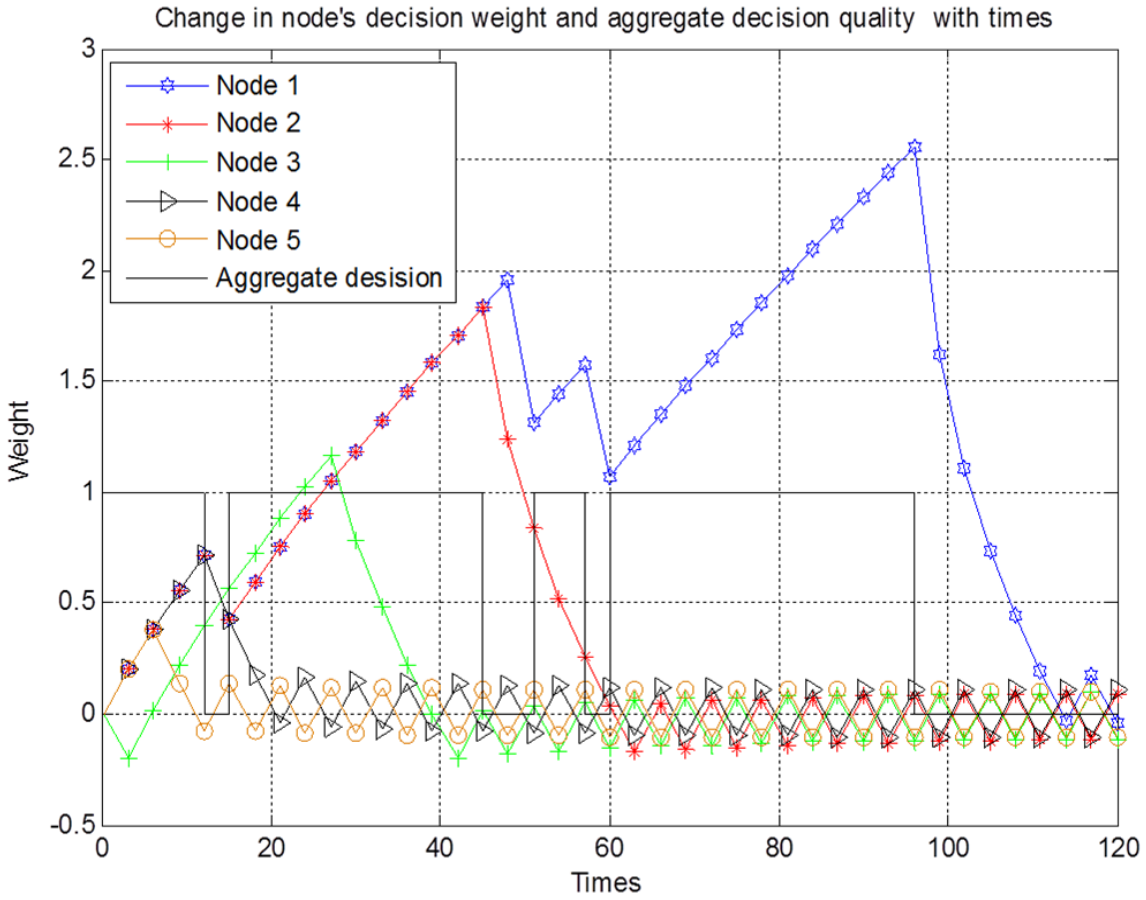

Figure 4: Repeated Game Simulation with Five Nodes

We can see in this simulation that, from time 60 to 93, the aggregate decision remains correct although 4 out of 5 nodes are compromised. The aggregate decision actually survives until the last node is compromised. Thus, the mission survival time is substantially improved because the attacker is denied the "multiplier effect" of fault tolerance lent by majority rule. This substantiates a fight-through OODA loop, enabled by multi-dimensional redundancy, that completes before that attacker can complete his OODA loop. The result: the attacker sees that the outcome of his successful attack is not a precipitous loss of the defender's resources once a majority is compromised; instead there is a dogged sustaining of targets. The increase in the perceived effort to sustain the attack thus serves to overwhelm the attacker. The defender having been afforded sufficient surviving resources has fought through the attack.

Replicas support mission survival; yet to the mission, the replicas should run transparently. A replicated process, for example, will produce replicated outputs that have to be resolved to a single output. Voting among the replicas resolves the multiple outputs and can prevent some malicious replicas from corrupting the outcome. Replication coupled with voting can therefore be a pervasive element of survival. The research in [25], [26] has compared two voting configurations: hierarchical voting (Figure 5) and simple majority.

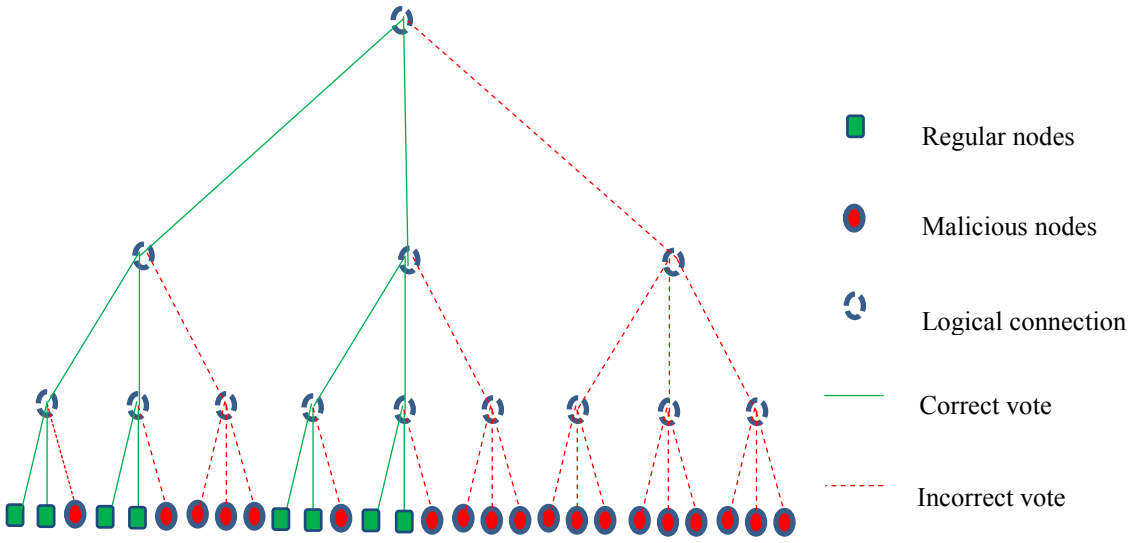

Figure 5: Hierarchical Troika Nodal Arrangement

Hierarchical voting is appealing because its similarity to "divide-and-conquer" expedites the voting outcome and admits voting on a larger scale. This research affirmed that a simple majority configuration, in spite of the speed and scalability of a hierarchical one, is superior in terms of withstanding attacks from compromised replicas within the voting configuration. This result is timely because it synergizes with recent rethinking [27] of how warfighting information should flow. Information has predominately been collected and pushed down through subordinates and eventually to the lowest level. Now this information pyramid is being inverted: the lowest, most populated level is being "elevated" so that it is the focus for making game-changing decisions. Technology developments are growing, and the future networked battlefield may see the lowest unit equipped for improving both the receipt and collection of tactical data. Shifting replica-based survivability to accommodate this change means enabling a versatile mix of configurations to fit the warfighter's need.

3.7 Trusted OSN (Online Social Network) Services

With the growing number of Internet users, Online Social Networks (OSNs) have become an important mode of communications, establishing a connection between people and providing a platform for online interaction. OSN sites have evolved as the highest growing medium

with expanded influence, involving various online activities on one platform for the people who are willing to communicate and share their resources with others. Technically, those OSN sites have integrated the functions of existing online services under one shelter, thus providing a more interesting and productive platform to the users.

An OSN service can provide numerous advantages to users and organizations by helping people to interact with each other and share their resources instantly, but it also introduces new challenges and vulnerabilities in terms of security and privacy. This is a critical challenge to both service providers and users, considering the numbers of users growing dramatically every day, because inappropriate usage of OSNs may cause privacy damages or security violations. The security and privacy issues in OSNs become major concerns that hinder the widespread adoption of OSN services – especially in sensitive organizations.

In general, survivability in a mission-critical system can be achieved by a three-pronged strategy: Prevention, Detection/Response, and Recovery [28]. According to this strategy, we must first of all attempt to prevent possible vulnerabilities and attacks from compromising mission-critical systems and resources. Undertaking this approach, we enhance the level of prevention in the highest-growing communication medium - OSN services. Ultimately, our trust mechanisms will make the current OSNs more survivable so that even sensitive organizations can rely on the services and increase their productivity and profit via reliable resource sharing with heterogeneous platforms, security, high availability, and cost-effective maintenance.

Although users are constantly warned about the security and privacy threats involved with the use of OSN sites, yet many ordinary users are unaware of the risks through the service. Furthermore, the third party applications, which are integrated with an OSN, may trigger security vulnerabilities or weaknesses to the entire OSN service.

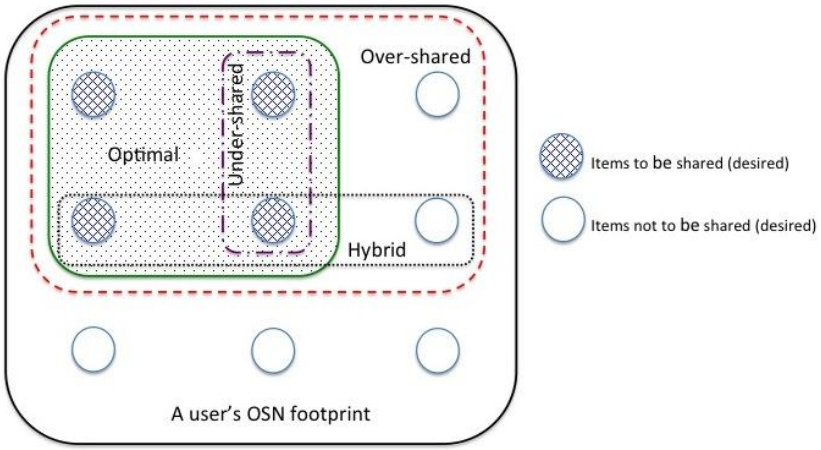

Figure 6: States of Data Sharing on OSNs

Currently, however, there are no standard countermeasures governing the OSN providers with regard to security, privacy, or reliability. As a result, each provider has full discretion over how they will manage these important issues, but they do not have strong technical solutions for those concerns. Due to the lack of reliable trust mechanisms in the current OSN services, sensitive organizations such as government agencies and DoD have not adopted the services in their environments yet, while there are various advantages demonstrated in other communities.

Therefore, in order to provide the same benefits of OSN services to we have analyzed the security and privacy vulnerabilities in OSN services and developed the support mechanisms for trusted OSN services, so that more users and sensitive organizations may adopt the trusted OSN services in their computing environments. The outcomes of this research should serve as a foundation for further research that will uncover solutions to trusted OSNs and lead to increased usage of the OSN services with its advantages. We believe the trusted OSNs will promote information superiority at sensitive organizations including IA community by providing effective trust mechanisms to a collection of universal information services that can be rapidly tailored and dynamically orchestrated to accommodate a variety of enterprise needs.

In particular, we have developed a framework that can provide trusted data management in

OSN services [29-32]. In summary, we first identified the data types on OSNs and the states of shared data, considering the desired and actual levels of data sharing, with respect to Optimal, Under-shared, Over-shared, and Hybrid (depicted in Figure 6). We then defined and analyzed the parameters that facilitate or detract from the level of data sharing on OSNs. Furthermore, we investigated the preventive parameters that help OSN users and service providers properly maintain sensitive information on OSNs, reducing the possibility of state transition to non-Optimal states.

In a reliable OSN service, users should be able to set up the desired levels of information sharing with specific groups of other users. However, it is not clear to an ordinary user how to decide how much information should be revealed to others. Therefore, in this research we proposed an approach for helping an OSN user determine his optimum level of information sharing, taking into consideration the payoffs (potential Reward or Cost) based on the Markov Decision Process (MDP).

The MDP-based approach can be considered as a one-player game, where the user is viewed as playing the game against impersonal background situation. As an extension of the MDP-based approach, we also introduced a game theoretic approach for helping OSN users to determine their optimum policy on OSNs in terms of data sharing, based on a two-player (i.e., user and attacker) zero-sum Markov game model [31]. This is a generalized framework of the MDP-based approach that considers the interactions of typical players in OSN services with conflicting interests whose decisions affect each other's. After developing the game-theoretical model, we conducted various attack simulations and discussed the results, considering random attackers without knowledge about the target, attackers with limited knowledge about the target, attackers with full knowledge about the target, and the risks of rapid public exposure [32].

Our research addresses the security and privacy problems in existing OSNs that have hindered the broader application of large-scale resource-sharing services in a sensitive organization. Our research outcomes offer an array of advanced approach for data-sharing in large-scale online computing services, developing new support mechanisms. We expect that

the trusted OSN services will enable the users to protect their security and privacy on the Internet and organizations to increase their productivity and profit via reliable resource sharing with heterogeneous platforms, security, high availability, and cost-effective maintenance.

From investigating OSNs we gained exposure to a technology that is a microcosm of the Internet and mobile media. We likewise devoted some of our attention to two other technologies that constitute the cyber infrastructure: wireless sensor networks and cognitive radio networks. Delving into the cognitive the radio paradigm is a meaningful arena for FTFT because cognitive radios pose challenges that are pervasive across cyberspace. Cognitive radios, due to their openness, make them both an enabler and a danger. Wireless sensor networks are also relevant to FTFT because they, like fault-tolerant computers, are often deployed in hostile environments where they have to contend with resource-limiting circumstances. In turning our attention to wireless sensor networks and cognitive radios we sought to introduce performance improvements to their role in cyberspace instead of elevating their assurance. Usually replication and diversity, as we have presented them for fault tolerance for fight through, are introduced after a component's functionally is implemented. While not explicitly showing how, we contend that our proposed designs OSNs, wireless sensor networks and cognitive radios will, if adopted, facilitate the use of replication and diversity for fighting-through.

3.8 Dynamic Reconfigurable Routing for Wireless Sensor Networks

The primary purpose of any infrastructure-less wireless sensor networks (WSNs) is to sense the environment and while doing so, they can deliver a multitude of services, ranging from reliable sensing, real time streams, mission critical support, network reprogramming and so on. The very nature of these networks calls for multi-hop wireless routing where all sensor nodes somehow decide on *how, when,* and *whom* to route the packets. Obviously, no one routing mechanism would suffice for the variety of tasks a sensor network performs—sending periodic sense-and disseminate flows, real time streams, mission critical alerts,

network reprogramming data, patched updates, interactive queries, commands and so on. In this regard, we have developed a dynamic reconfigurable routing framework for wireless sensor networks [33].

The main idea behind the dynamic reconfigurable routing framework is the exposition of the flow's requirement using just 3 bits in the packet header. The 3 bits in the header represent i) whether the packet is a control packet or a data packet, ii) if the packet is delay sensitive, and iii) if the packet needs reliability. The inferences due to the 8 possible combinations of these bits are shown in the Table 2.

Table 2: Combination of Preamble Bits and Their Inferences

Data (0) Control (1)	Real-time (1)/ Non-real-time (0)	Reliable (1)/ Unreliable (0)	Inference
0	0	0	Unreliable, non real time, data packet
0	0	1	Reliable, non real time, data packet
0	1	0	Time critical, unreliable, data packet
0	1	1	Mission critical data packet
1	0	0	Unreliable, non real time, control packet
1	0	1	Reliable, non real time, control packet
1	1	0	Real time, unreliable, control packet
1	1	1	Mission critical control packet

The proposed dynamic routing framework (shown in Figure 7 as also seen in [33]) consists of a collection of routing components that optimize the routing for a given class of traffic. The suffix of the control packets (C0–C3) and data packets (D0–D3) indicates the status of the reliability and real time bits. For example, packet D3 would have preamble bits set to [0, 1, 1], with the first bit indicating a data packet and last 2 bits account for the suffix 3. Two virtual queues, one each for data and control traffic, take the incoming packets and schedule them for transmission to the lower layers of the stack. The shared neighbor table houses values such as the node-ID, energy available, congestion level, depth, link quality estimate and a 'last heard' bit. Since routing components share this table, it decouples core protocol features from interface assumptions and regularizes data structure requirements. This leaves the routing layer with a composable set of routing components that can be seamlessly ported across various research efforts.

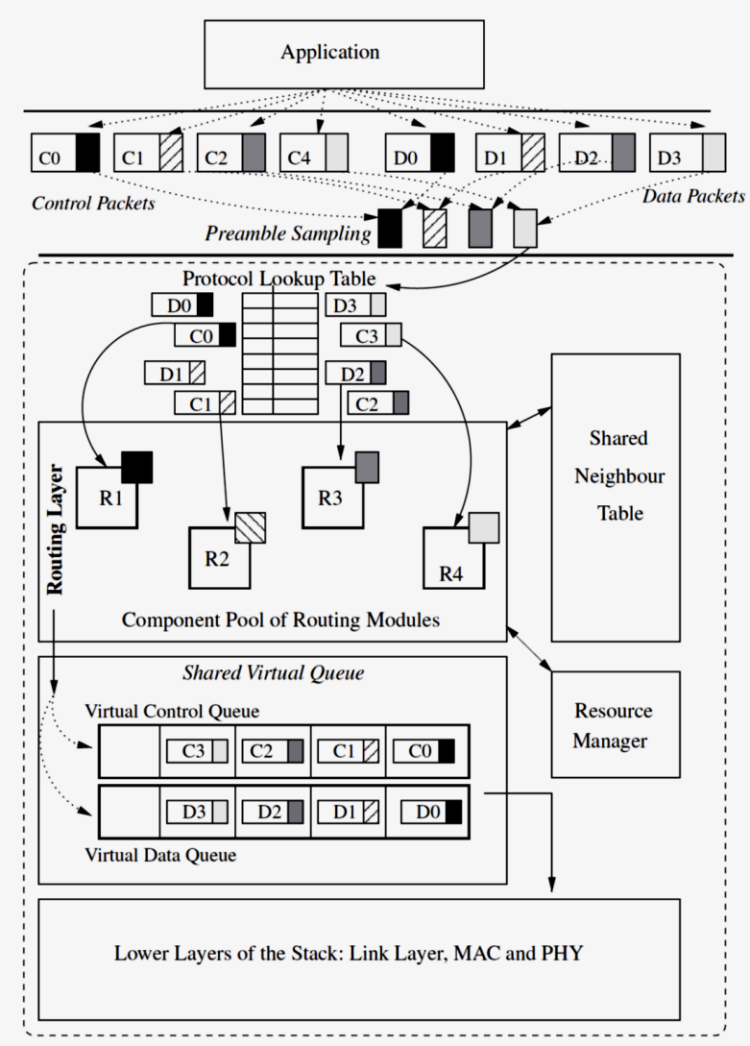

Figure 7: Dynamic Routing Framework

The evaluation of the framework was done via both simulations and testbed implementations. Predictive maintenance (PdM) was chosen as the target application to test the framework. PdM's variety of requirements of periodic reports, streaming real time values, query response, continuous customization, network reprogramming, and mission critical alerts were mapped to the 3 bit patterns. Realistic traffic scenario was generated for the PdM application. The routing framework was also implemented on a 40 node MicaZ wireless testbed with the nodes arranged in a rectangular 8 X 5 grid. Results revealed the differential routing capability of the proposed routing framework.

3.9 Bargaining for Radio Spectrum Sharing

With the Presidential Memorandum on Unleashing the Wireless Broadband Revolution [34], there has been a lot of recent activity on how to cater to the demands of wireless services in the years to come. Though some of the demands can be met with incremental technological advancements, a serious imbalance in the supply and demand still looms. This calls for a clean-slate design not only on the radio engineering solutions but also on how the market must evolve so that the resources are more efficiently utilized for better service offerings with competitive pricing. The most vital resource for any wireless application or service (i.e., mobile telephony, TV and radio broadcasts, GPS, maritime navigation) is the radio spectrum. Thus, it becomes absolutely essential that the available radio spectrum is utilized in the best possible manner.

Spectrum allocation and management have traditionally followed a 'command and control' approach – regulators like the Federal Communications Commission (FCC) allocate spectrum to specific services under restrictive licenses. These limitations have motivated a paradigm shift from static spectrum allocation towards a more 'liberalized' notion of dynamic spectrum management in which secondary networks/users (non-license holders) can `borrow' and `share' idle spectrum from those who hold licensees (i.e., primary networks/users), without causing harmful interference to the latter – a notion commonly referred to as the dynamic spectrum access (DSA). The most hotly debated question is: "how should spectrum be allocated and managed" by the authorities? The general consensus is to allow multiple secondary networks to compete and co-exist for better utilization of radio resources.

In this regard, we have developed a bargaining framework that takes into consideration the interference from other networks such that all networks can co-exist. A preliminary version of this work was presented in IEEE WCNC 2012 in Paris, France [35]. The complete work is currently under review in the *IEEE/ACM Transactions on Networking* [36].

Our main contribution lies in proposition of a bargaining framework for radio spectrum

sharing. We consider nodes to behave in a selfish manner, i.e., each node solely focuses on maximizing its utility by accessing and using as many channels as possible from the set of orthogonal channels not currently being used by any of the primary incumbents. The nodes in our model, for example, can correspond to broadcast access points deployed by competing wireless service providers. By using more channels each provider may intend to support more customers for maximizing its revenue. The channels that a node selects is, however, subject to the following constraint– nodes within the interference range of each other have to use orthogonal channels to minimize interference. Thus, the nodes will have to agree upon a *sharing rule* of the channels among themselves, i.e., each node will have to decide "how many" and "which" channels to use. In other words, *the channel access problem by a set of selfish nodes is inherently a bargaining game*. The fundamental question that we address is– *how many* and *which* channels each node should access to maximize its gain. Specifically, we model the problem of agreeing upon a sharing rule of the channels among the nodes as an infinite horizon Rubinstein-Stahl bargaining game. In our model, each node "bargains" with the other nodes (opponents) in the network regarding its "share" (*how many* and *which*) of the channels. Notice that, until the nodes agree upon the sharing rule, none of nodes can start data communication. Thus, "waiting" for the bargaining outcome also costs the nodes. We consider this cost by discounting future payoff of the nodes. This discounting represents the *patience* of the nodes in waiting for the bargaining outcome. We argue that it is the relative patience of the players that influence the degree of fairness in the sharing rule. We show more patient players tend to get a larger fraction of available channels.

We solve the bargaining game by deriving Subgame Perfect Nash Equilibrium (SPNE) strategies of the players in the game. The SPNE strategies that we derive comprise a set of strategies such that, no player in a subgame can deviate from these strategies and thereby gain from his deviation. We investigate finite horizon version of the game and identify its SPNE strategies. We then extend these results to the infinite horizon bargaining game. We provide polynomial time algorithms to find the SPNE strategies of both the finite and infinite horizon versions of the game. Furthermore, we identify Pareto optimal equilibria of the game for improving spectrum utilization. We also conduct simulations to study how the self-gain maximizing strategies of the players affect system wide performance.

The proposed solution concept can be best described using an illustrative example with 6 nodes as shown in Figure 8. The graphs in the figure depict the conflict graph of the network. The number of channels available, M, has been assumed to be 5. Each node has a discount factor of 0.5. The game is played for 6 periods. The channels assigned to the nodes in each period have been shown in brackets beside the node.

Our objective is to find the SPNE strategy of $P1$ in the first period of the game. Figure 8(a) shows the Pareto optimal NE strategy of $P6$ in the last period of the game. The offerer in this period is $P6$. First, a NE strategy of $P6$ in the last stage (which need not be Pareto optimal) is found. $P6$ colors his neighbors ($P2$ and $P3$) with the least possible number of colors (channels) and keeps rest of the channels for himself (thereby maximizing his share). The neighboring nodes (i.e., $P1$, $P4$ and $P5$) are given a channel each by graph coloring them. This is done by considering $P1$, $P4$ and $P5$ in non-increasing order of their degree in the subgraph induced by $P1$, $P2$, $P3$, $P4$ and $P5$. Thus, $P1$ is considered first and gets $C2$, next $P5$ gets $C3$ and finally $P4$ gets channel $C1$. Clearly, the channel assignment obtained so far corresponds to a NE strategy of $P1$ in the last stage, but one that may not be Pareto optimal, since the shares of some players ($P1$, $P4$ and $P5$) can be improved without hurting the share of any other player.

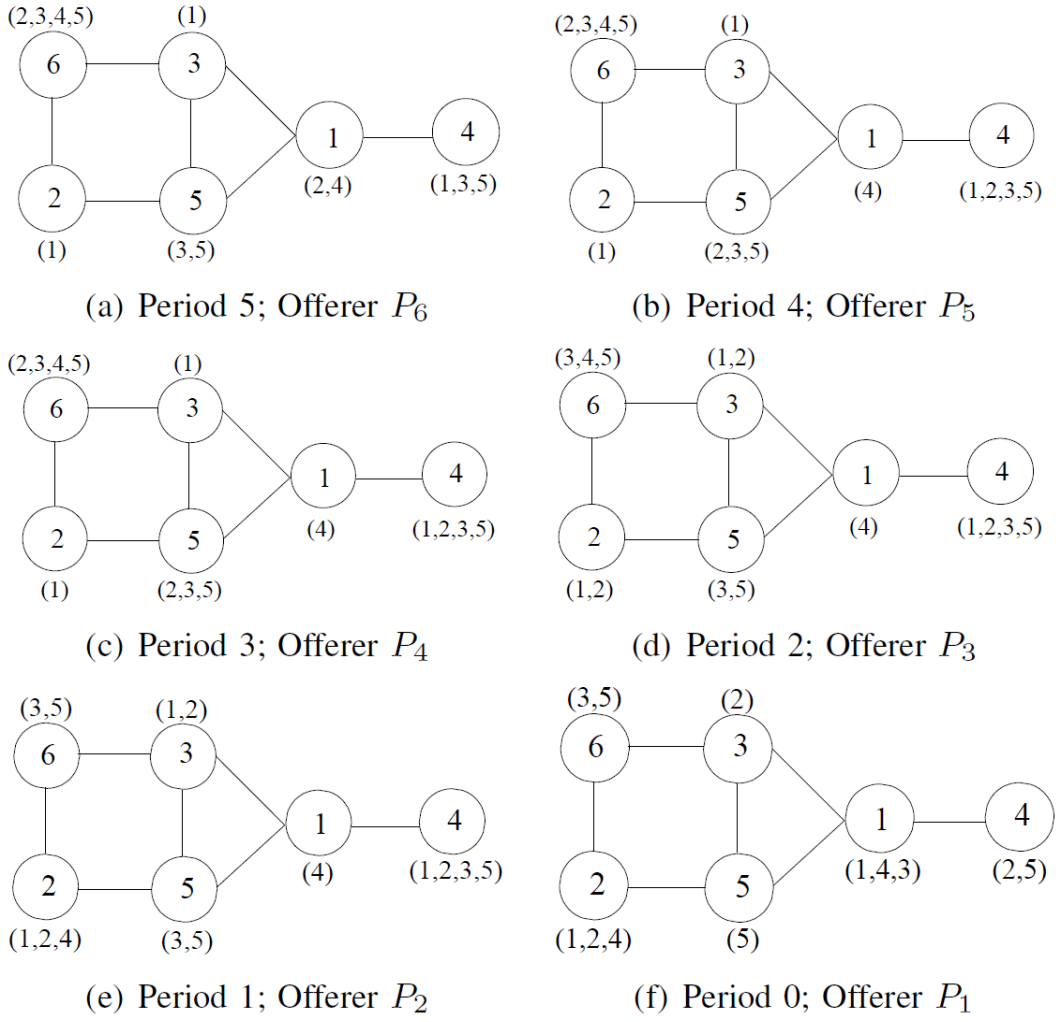

Figure 8: Pareto Optimal SPNE of the Respective Offerer in Different Periods

The Pareto optimal NE strategy of *P*6 is obtained by considering the players in *P-6* and checking to see if more channels can be assigned to the player. In the first iteration, *P*1 receives *C*4, *P*2 and *P*3 does not get any more channels, *P*4 gets *C*3 and *P*5 gets *C*5. In the second iteration only *P*4 gets *C*5. The channel assignment obtained now is shown in Figure 8(a), and corresponds to the Pareto optimal NE strategy of *P*6 in the last stage of the game.

Following the same line of reasoning, we finally obtain the SPNE strategy of *P*1 in the first period of the game. This strategy of *P*1 is shown in Figure 8(f).

The proposed spectrum bargaining is fundamentally different and much more difficult than conventional Rubinstein-Stahl bargaining. This is because of two primary reasons– (i) spectrum can be spatially reused concurrently; two conflicting players must not use the same channels simultaneously yet well-separated players can, and, (ii) players can only use whole channels, not fractional channels. We consider both constraints while analyzing the spectrum bargaining game.

4.0 Results and Discussion

Fault tolerance is an important component of cyber defense. In fact, fault tolerance deals with detection and treatment of failures and cyber defense deals with detection and treatment of violations. A well-formed and well-founded framework for treating violations comes from fault-tolerant computing. Traditionally, fault-tolerant computing dealt with randomly occurring faults and not faults resulting from intelligent attack. However, a reliable system should be survivable against the faults caused by cyber attacks as well as the internal failures. Whereas faults caused by natural-occurring phenomena are tolerable using established, standard approaches, attacker-induced faults require a more aggressive approach.

The results from the individual tasks described above demonstrate the aggressiveness needed to transform fault tolerance to a fight through capability. Nonetheless, new challenges arise in the area of transforming fault tolerance to attack tolerance. We have witnessed some of these challenges in our investigations into OSNs. Our choosing of OSNs for an area of investigation was motivated by how OSNs are changing the world as part of cloud computing.

Cloud computing has attracted users seeking the cloud's ubiquity, convenience, and on-demand network accessibility. Although promoting availability, the cloud's perceived vulnerabilities have spurred researchers to find ways to assure availability in the cloud [37-39]. Unavailability is not the only possible detriment to adopting the cloud computing model; integrity violations also became the concern of researchers [40], [41]. Of particular

interest to us is the proposed use of concepts from the domain of fault-tolerant computing to address availability and integrity in cloud computing. The deliberate introduction of redundancy in the cloud's provisioning [42] underscores the suitability of cloud computing as a target for fault tolerance concepts. Unlike random, naturally-occurring faults, the aggressiveness with which attackers induce faults has motivated researchers to propose schemes that make fault tolerance more robust.

Although none of these issues is new in the world of computing, compared with traditional infrastructures, cloud computing architectures exhibit a different partitioning with respect to security and survivability issues [43-46].

The cloud's attributes of ubiquity, convenience, and on-demand network accessibility served as a springboard for us to examine some of the cloud's outermost reaches: wireless sensor networks and cognitive radios. The two technologies, while not part of the cloud computing concept, are tied to the movement of data to and from the cloud; therefore they become key ingredients to the cyberinfrastructure and hence were deserving of our attention.

Researchers [47], [48] have expanded upon traditional fault tolerance to make it robust enough to withstand attack. Using calculations of the mean-time-to-compromise [3], [4], these schemes employ sufficient amounts of resource redundancy in the framework of a general distributed system to withstand aggressively-created faults. They do not, however, achieve a comprehensive path to survivability that, in addition to tolerating, will avoid, prevent, and recover from faults. As a result, these schemes limit themselves by not fully exploiting the fault fighting features that could be realized in cloud computing. We contend that the approaches adopted by FTFT to transform fault tolerance to a fight through capability are amendable to address some of the main security challenges faced by cloud computing.

Finally, although the results from our investigations into wireless sensor networks and cognitive radios did not deal with fault tolerance per se, they nonetheless contributed to the closely-related issue of dealing with resource constraints. The approaches that we proposed

in wireless sensor networks and cognitive radios are proactive whereas traditional fault tolerance is, for the most part, reactive. In a larger sense, our transforming fault-tolerant computing concepts into a fight through capability, is a blending of the reactive and the proactive: our proposed fight through OODA Loop is reactive to faults yet strives to proactively anticipate the attacker's optimal next action. Conceptually, this results in concentric OODA Loops as depicted in Figure 9. The outer loop is the attacker's OODA Loop and the inner loop belongs to the defender. The tighter diameter of the defender's OODA Loop illustrates the defender's more timely completion of the cycle. FTFT's multi-dimensional redundancy permits the defender to "get inside the enemy's decision cycle."

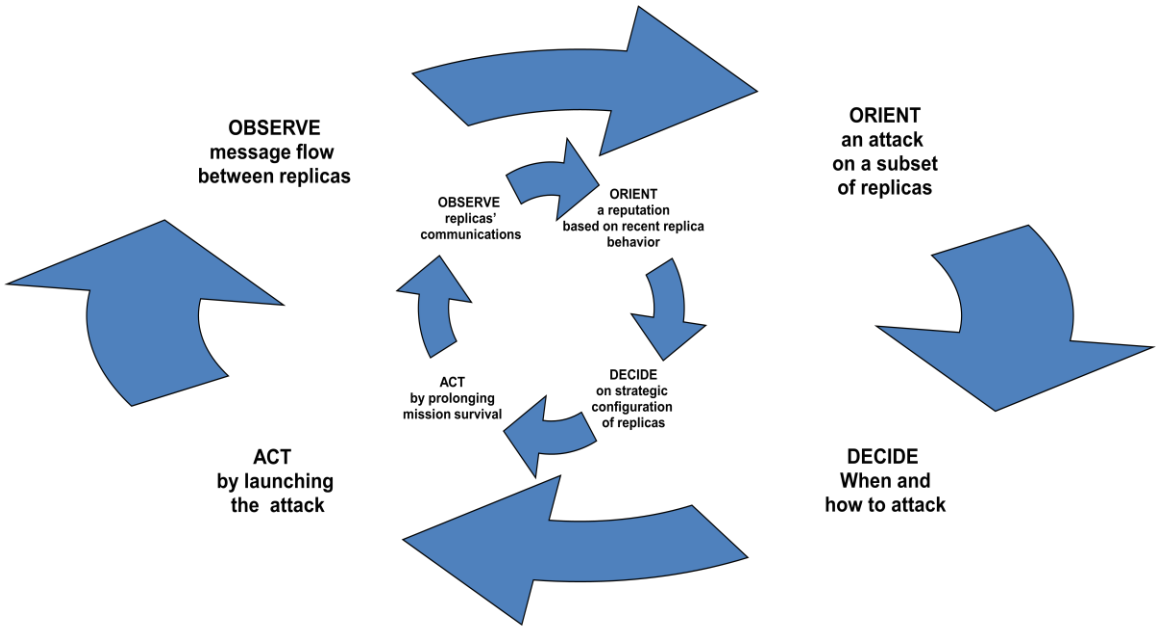

Figure 9: Concentric OODA Loops: Attacker's Outer Loop, Defender's Inner Loop

5.0 Conclusions

FTFT offers discriminators from previous endeavors to apply fault tolerance for cyber defense. By simultaneously combining spatial, temporal and information redundancy we form a triad that establishes an OODA loop for fight-through. Recall that MTTB and MTTC are coarse-grain measures applied at network design time for configuring firewalls, hardening servers, and

placing intrusion sensors in anticipation of attacks. We proactively and propitiously adapt to undermine the attacker's knowledge of our fault-tolerant system. Our adaption of fault tolerance techniques is multi-dimensional: an attacker inflicting punishment upon the fault tolerance mechanisms of one dimension will not defeat the entire fight-through capability; instead, the other dimensions become the reinforcements for fighting through. To defeat the triad, the attack must succeed against all the dimensions simultaneously.

We demonstrated how to stand-up multidimensional redundancy to increase the attacker's target space. In so doing, we force the attacker to attempt overtaking every dimension of redundancy simultaneously. This can be decisive: it permits the continual defeat of an adversary's optimal strategies.

6.0 References

1. Frans P.B. Osinga, *Science, Strategy and War: The Strategic Theory of John Boyd*, Routledge Publishing, 2006.

2. *OASIS: Foundations of Intrusion Tolerant Systems*, Jaynarayan H. Lala editor, IEEE Computer Society Press, 2003.

3. http://www.spacedaily.com/news/cyberwar-04g.html

4. David Leversage and Eric Byres, "Estimating a System's Mean Time-to-Compromise," *Journal of Security and Privacy*, Vol. 6, Issue1, IEEE, 2008.

5. Erland Jonsson and Tomas Olovsson, "A Quantitative Model of the Security Intrusion Process Based on Attacker Behavior," *IEEE Transactions of Software Engineering*, Vol. 23, No. 4, 1997.

6. Barry Johnson, *Design and Analysis for Fault-Tolerant Digital Systems*, Addison-Wesley Publishers, 1989.

7. Kevin Kwiat, Alan Taylor, William Zwicker, Daniel Hill, Sean Wetzonis, and Shangping Ren, "Analysis of Binary Voting Algorithms for use in Fault-Tolerant and Secure Computing," *IEEE Proceedings of the International Conference on Computer Engineering and Systems (ICCES),* Cairo, Egypt, December 2010, pp. 269-273.

8. Roger Myerson, *Game Theory: Analysis of Conflict*, Harvard University Press, 1997.

9. Li Wang, Zheng Li, Shangping Ren, and Kevin Kwiat, "Optimal Voting Strategy Against Rational Attackers," *Proceedings of the International Conference on Risk and Security of Internet and Systems (CRISIS)*, 2011, pp. 1–8.

10. Li Wang, Shangping Ren, Ke Yue, and Kevin Kwiat, "Optimal Resource Allocation for Protecting System Availability Against Random Cyber Attacks," *Proceedings of the International Conference on Computer Research and Development (ICCRD),* 2011 , vol. 1,

pp. 477–482.

11. Li Wang, Yair Leiferman, Shangping Ren, and Kevin Kwiat, and Xiaowei Li, "Improving Complex Distributed Software System Availability Through Information Hiding," *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp. 452–456.

12. Mohammad Rabby, Kaliappa Ravindran, and Kevin Kwiat, "Hierarchical Adaptive QoS Control for Voting-based Data Collection in Hostile Environments," *Proceedings of the 8th International Conference on Network and Service Management (CNSM)*, 2012, pp. 194-198.

13. Eldon C. Hall, *Journey to the Moon: The History of the Apollo Guidance Computer*, American Institute of Aeronautics & Astronautics, 1996.

14. Edgar Ulsamer, "Computers – Key to Tomorrow's Air Force," *Air Force Magazine*, July 1973, pp. 46-52.

15. Peter A.Lee and Thomas Anderson, *Fault Tolerance: Principles and Practice*, Second Revised Edition, Springer-Verlag, 1990.

16. Brian Randell, "System Structure for Software Fault Tolerance," *IEEE Transactions on Software Engineering*, Vol. SE-1, No. 2, June 1975, pp. 220-232.

17. Algirdas Avizienis, "The *N*-Version Approach to Fault-Tolerant Systems," *IEEE Transactions on Software Engineering*, Vol. SE-11, No. 12, December 1985, pp. 1,491-1,501.

18. Jean-Claude Laprie, Jean Alrat, Christian Be'ounes, and Karama Kanoun, "Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures," IEEE *Computer*, July 1990, pp. 39-51.

19. Alfred Spector, and David Gifford, "The Space Shuttle Primary Computer System," *Communications of the ACM*, Vol. 27, No. 9, September 1984, pp. 874-901.

20. Ken Birman, and Fred Schneider, "The Monoculture Risk Put Into Context," *IEEE*

Security and Privacy, Vol. 7, No. 1, January/February 2009, pp. 14-17.

21. Russell J. Abbott, "Resourceful Systems for Fault Tolerance, Reliability, and Safety," *ACM Computing Surveys*, Vol. 22, Issue 1, March 1990, pp. 35-68.

22. Vincent Rahli, Nicolas Schiper, Robbert van Renesse, Mark Bickford, Robert L. Constable, "A Diversified and Correct-by-Construction Broadcast Service," *Proceedings of the 2nd International Workshop on Rigorous Protocol Engineering (WRiPE),* October 2012, pp. 1-6.

23. Charles Kamhoua, Kevin Kwiat, Joon Park "Surviving in Cyberspace: A Game Theoretic Approach" in the *Journal of Communications, Special Issue on Future Directions in Computing and Networking,* Academy Publisher, Vol. 7, No 6, June 2012.

24. Ellison, G. "Cooperation in the Prisoner's Dilemma with Anonymous Random Matching" *The Review of Economic Studies*, Vol. 61. No. 3, July, 1994, pp. 567-588.

25. Charles Kamhoua, Patrick Hurley, Kevin Kwiat, Joon Park "Resilient Voting Mechanisms for Mission Survivability in Cyberspace: Combining Replication and Diversity" in the *International Journal of Network Security and Its Applications (IJNSA),* Vol.4, No.4, July 2012, pp. 1-19.

26. Charles Kamhoua, Kevin Kwiat, Joon Park "A Binary Vote Based Comparison of Simple Majority and Hierarchical Decision for Survivable Networks" in the *Proceedings of the Third International Conference in Communication Security and Information Assurance (CSIA 2012)*, Published by Springer, Delhi, India, May 2012, pp. 883-896.

27. "Inverting the Information Pyramid," *Federal Computer Week*, Vol. 26, No.4, March 30, 2012. Available online at: http://www.defensesystems.com/C4ISRreport

28. Joon S. Park, Pratheep Chandramohan, Avinash T. Suresh, Joseph Giordano, and Kevin Kwiat. "Component Survivability for Mission-Critical Distributed Systems," Special Issue on Cloud and Pervasive Computing, *Journal of Supercomputing*, 2012. In press (online version is available at http://www.springerlink.com/content/510226706112j753/)

29. Joon S. Park, Sookyung Kim; Charles Kamhoua, and Kevin Kwiat, "Towards Trusted

Data Management in Online Social Network (OSN) Services," *World Congress on Internet Security (WorldCIS)*, June 10-12, 2012., pp.202-203.

30. Joon Park, Sookyung Kim, Charles Kamhoua, and Kevin Kwiat, "Optimal State Management of Data Sharing in Online Social Network (OSN) Services," *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom),* June 25-27, 2012, pp. 648-655.

31. Charles Kamhoua, Kevin Kwiat, Joon S. Park, "A Game Theoretic Approach for Modeling Optimal Data Sharing on Online Social Networks," *9th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE),* September 26-28, 2012, pp. 271-276.

32. Jonathan White, Joon S. Park, Charles Kamhoua, Kevin Kwiat, "Game Theoretic Attack Simulation in Online Social Network (OSN) Services," *IEEE Intelligence and Security Informatics (ISI)*, June 4-7, 2013. In progress.

33. Mukand Venkataraman, Mainak Chatterjee and Kevin Kwiat, "A Dynamic Reconfigurable Routing Framework for Wireless Sensor Networks," *Elsevier Journal on Ad hoc Networks*, Vol. 9, Issue 7, Sept. 2011, pp. 1270-1286.

34. http://www.whitehouse.gov/the-press-office/presidential-memorandum-unleashing-wireless-broadband-revolution

35. Swastik Brahma and Mainak Chatterjee, ``Spectrum Sharing in Secondary Networks: A Bargain Theoretic approach", IEEE Wireless Communications and Networking Conference (WCNC), 2012, pp. 1331-1336.

36. Swastik Brahma, Mainak Chatterjee, Kevin Kwiat, ``Interference Aware Bargaining Framework for Self-Coexistence in Unlicensed Spectrum", Under review in *IEEE/ACM Transactions on Networking,* 2012.

37. Juan Du, Xiaohui Gu, Douglas Reeves, "Highly Available Component Sharing in Large-Scale Multi-Tenant Cloud Systems," *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC),* June 2010, pp. 85-94.

38. Bernardetta Addis, Danilo Ardagna and Barbara Panicucci, and Li Zhang, "Autonomic Management of Cloud Service Centers with Availability Guarantees," *Proceedings IEEE 3rd International Conference on Cloud Computing (CLOUD),* July, 2010, pp. 220-227.

39. Song Fu, "Failure-Aware Resource Management for High-Availability Computing Clusters with Distributed Virtual Machines," *Journal of Parallel and Distributed Computing* Vol. 70, No. 4, April, 2010, pp. 384-393.

40. Juan Du, Wei Wei, Xiaohui Gu, Ting Yu, "RunTest: Assuring Integrity of Dataflow Processing in Cloud Computing Infrastructures," *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACC),* 2010, pp. 293-304.

41. Juan Du, Nidhi Shah, Xiaohui Gu, "Adaptive Data-Driven Service Integrity Attestation for Multi-Tenant Cloud Systems," *Proceeding of the IEEE 19th International Workshop on Quality of Service (IWQoS),* June, 2011, pp. 1-9.

42. Sebastian Stein, Terry R. Payne, and Nicholas R. Jennings, "Robust Execution of Service Workflows Using Redundancy and Advance Reservations," *IEEE Transactions on Services Computing,* Vol. 4, No. 2, 2011, pp. 125-139.

43. Joon S. Park and Jerry Robinson, "Security Mechanisms for Trusted Cloud Computing," Proceedings *of the International Conference on Cloud Computing & Virtualization (CCV)*, Singapore, May, 2010.

44. Bernd Grobauer, Tobias Walloschek, and Elmar Stocker, "Understanding Cloud Computing Vulnerabilities," *IEEE Security and Privacy* Vol. 9, No. 2, March 2011, pp. 50-57.

45. Hassan Takabi, James B. D. Joshi, and Gail-Joon Ahn, "Security and Privacy Challenges in Cloud Computing Environments," *IEEE Security and Privacy* Vol. 8, No. 6 November, 2010, pp. 24-31.

46. Siani Pearson and Azzedine Benameur, "Privacy, Security and Trust Issues Arising from Cloud Computing," *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science* (CLOUDCOM '10), November, 2010, pp. 693 - 702.

47. Kjell Hausken, "Strategic Defense and Attack for Series and Parallel Reliability Systems," *European Journal of Operational Research*, Vol. 186, No. 2, April 2008, pp. 856–881.

48. Gregory Levitin and Kjell Hausken, "False Targets vs. Redundancy in Homogeneous Parallel Systems," *Reliability Engineering & System Safety,* Vol. 94, No. 2, 2009, pp. 588–595.